6М

УДК 004.896:62-523.8

В.П. Макарычев
Государственный научный центр Российской Федерации, Центральный
научно-исследовательский институт робототехники и технической кибернетики,
г. Санкт-Петербург, Россия
makar@.rtc.ru

Метод переменных стратегий построения траекторий движения роботов в среде с препятствиями

В работе предложен и исследован метод построения траекторий движения роботов в среде с препятствиями, основанный на смене стратегий поведения в зависимости от ситуации. Метод состоит из ряда шагов. Сначала делается попытка построения в целевую точку канонической траектории в свободном пространстве, оптимальной по быстродействию и с учётом динамических ограничений. Быстрый анализ наличия пересечений построенной траектории с препятствиями основан на их пирамидальной структуре. Описаны реализующая разработанный метод компьютерная программа на Matlab и результаты экспериментального исследования на примере макета космического технологического манипулятора.

Существует многообразие алгоритмов и методов построения траекторий движения для роботов: мобильных и манипуляторов [1], [2]. Однако до сих пор отсутствует единый универсальный подход к построению траекторий в среде с препятствиями.

Можно выделить три группы методов планирования пути (траектории как геометрической кривой).

1. Методы потенциальных функций.
2. Клеточно-графовые методы.
3. Методы вероятностной дорожной карты.

В методах потенциальных функций построение пути робота в заданную точку осуществляется за счет действующих на него потенциальных сил двух типов: силы притяжения к цели и сил отталкивания от границ препятствий. Для создания поля этих сил искусственно формируются потенциалы, и решается экстремальная задача без ограничений. Главным недостатком метода потенциальных функций является его громоздкость в случае многомерных пространств и сложность потенциальной функции: наличие локальных экстремумов.

Более гибкими логическими возможностями при обходе известных препятствий теоретически любой сложности обладают клеточно-графовые методы, использующие описания проблемной среды в виде клеток и соответствующих им графов. Далее ищется оптимальный путь на графе [3], что в случае большой размерности может быть трудным.

Современный популярный метод вероятностной дорожной карты (PRM – Probabilistic Road Map) [4], [5] показал свою эффективность и преимущества по сравнению с рассмотренными выше классическими методами. Сначала на фазе планирования строится дорожная карта с помощью повторяемого алгоритма построения случайных конфигураций, и в качестве «узлов» дорожной карты выбираются свободные, т.е. не пересекающие препятствия, конфигурации. Затем находятся траектории, соединяющие выбранные узлы с помощью простого и быстрого алгоритма, который лежит в основе процедуры планиро-

вания пути PRM и называется локальным планировщиком. Таким образом, формируется граф, в котором конфигурации являются узлами графа, а вычисляемые локальным планировщиком пути – дугами.

На фазе исполнения решается задача нахождения пути между двумя свободными конфигурациями робота на карте аналогично клеточно-графовым методам. Итоговый путь получается дополнением траекторий из начальной и целевой точек в узлы графа.

Заметим, что случай заранее неизвестных препятствий сводится к случаю статических препятствий, когда в каждый момент времени имеется квазистатическая карта этих препятствий. Конечно, для эффективного решения задачи требуется, чтобы и метод построения траекторий был эффективным, т.е. быстрым.

Структура метода переменных стратегий

В работах [6-8] был предложен новый подход, который назовём «методом переменных стратегий», основанный на следующих компонентах:

– канонической процедуре построения субоптимальной по времени траектории в конфигурационном С-пространстве робота, с учетом кинематических и динамических ограничений, но без учёта препятствий;

– таблице известных безопасных (или свободных, т.е. не пересекающихся с препятствиями) траекторий, ранее построенных системой при фиксированном множестве статических препятствий;

– локальной коррекции траектории в случае пересечения канонической (п.1) траектории с препятствием и отсутствии траектории в таблице безопасных траекторий.

На рис. 1 изображена структура метода.



Рисунок 1 – Структура метода переменных стратегий

Изначально строится субоптимальная по времени, с учетом динамических ограничений, программная траектория в конфигурационном *C*-пространстве обобщенных координат Q^n, где n – число координат. Используется процедура, использующая параметризацию

траектории длиной дуги в этом римановом пространстве и подробно описанной в [7], [8]. Две точки $q^0, q^T \in Q^n$ соединяются прямой в евклидовой метрике Q^n и получаются аналитические выражения для её координат как функций параметра длины дуги

$$q(s) = q^0 + \frac{q^T - q^0}{\|\Delta q\|} s = q^0 + \frac{\Delta q}{\|\Delta q\|} s, \quad s \in [0, S], \quad \|\Delta q\| = S = \sqrt{\sum_{i=1}^{n} \left(q_i^T - q_i^0\right)^2}\ . \tag{1}$$

Разбивая траекторию на некоторое число узлов, например, равномерно по параметру времени t, $s^j = s^j\left(t^j\right)$, $\mathrm{j} = 0, 1, n_p$, $\mathrm{s}^0\left(t^0\right) = 0$, $\mathrm{s}^{\mathrm{np}}\left(\mathrm{s}^{\mathrm{np}}\right) = s(T) = S$, получаем множество узловых точек со значениями скоростей и ускорений в них

$$q^j(s) = q^0 + \frac{\Delta q}{\|\Delta q\|} s^j, \quad \dot{q}^j(s) = \frac{\Delta q}{\|\Delta q\|} \dot{s}^j, \quad \ddot{q}\ (s) = \frac{\Delta q}{\|\Delta q\|} \ddot{s}^j. \tag{2}$$

Представление препятствий

Образующие препятствия и робот геометрические тела представляются в виде пирамидальной структуры совокупности вложенных друг в друга и содержащих препятствие и робот простых тел (параллелепипед, шар, цилиндр) разного уровня [5]. Так, области на 1-м уровне содержат все препятствия и робот, области 2-го уровня содержатся в областях 1-го уровня и имеют меньшие размеры – они детализируют области 1-го уровня, и т.д. При анализе пересечений, если отсутствуют пересечения с какой-либо областью i-го уровня, то дальнейшее разбиение этой области i-го уровня не производится. Если же имеется пересечение с областью i-го уровня, то производится анализ пересечений для всех содержащихся в нем областей (i+1)-го уровня и т.п. Разбиение препятствий на области разных уровней, начиная с 1-го уровня, производится либо программистом-разработчиком, программистом-оператором, либо автоматически модулем СУД автоматической системы разбиения, имеющей черты искусственного интеллекта.

Препятствия опишем в виде областей $\widehat{G}_i \in W \subset R^m, i = 1, n_W$, в рабочей зоне W (также, области) евклидова пространства $R^m, m = 2{,}3$ для плоской или пространственной задачи, соответственно. Аналогичным образом опишем робот: $G_i = G_i(q) \in W \subset R^m$, $i = 1, n_R$, выделяя в его конструкции отдельные элементы G_i, рассматриваемые как абсолютно твердые тела (например, основание мобильного робота или звенья манипулятора). Тогда запретная зона совокупности препятствий имеет вид $\widehat{G} = \bigcup_{i=1,n_W} \widehat{G}_i \in W \subset R^m$, а робот – $G = G(q) = \bigcup_{i=1,n_R} G_i(q) \in W \subset R^m$.

Опишем разбиения областей на иерархические системы элементарных областей, являющихся покрытиями и вложениями областей (важный частный случай – описанные и вписанные окружности для достаточно симметричных областей, в особенности, если они близки). Области на уровне «*k*» будем обозначать с помощью k-индексов. Далее, $S_{i1} \subset S = S_0 = \bigcup_{i1=1}^{I1} S_{i1} \subset S, i_1 = 1, I_1$ – области 1-го уровня, содержащиеся в S_0; $S_{i1,i2} \subset S_{i1} = S_{i1,0} = \bigcup_{i2=1}^{I2} S_{i1,i2}, i_1 = 1, I_1, i_2 = 1, I_2$ – области 2-го уровня, содержащиеся в $S_{i1,0}$; $S_{i1,i2,\ldots,ik} \subset S_{i1,i2,\ldots,i(k-1)} = S_{i1,i2,\ldots,i(k-1),0} = \bigcup_{ik=1}^{Ik} S_{i1,i2,\ldots,i(k-1),ik}, i_1 = 1, I_1, i_2 = 1, I_2, \ldots, i_{k-1} = 1, I_{k-1}$ – области

k-го уровня, содержащиеся в $S_{i1,i2,...,i(k-1)}$. Здесь всюду индекс «0» обозначает все области следующего уровня, содержащиеся в проиндексированной до этого уровня области – как бы отсутствие дальнейшего разбиения. Кроме того, введем индексацию, при которой каждая область имеет k индексов, дополняя отсутствующие индексы нулями, а именно $S_{i1,i2,...,ij,0...0} = S_{i1,i2,...,ij,0} = S_{i1,i2,...,ij}$ и введем соглашение, что $S_{i1,i2,...,ik} = 0,\ \text{if}\ \exists \text{j}: i_j > \text{I}_\text{j}$.

Таким образом, получаем представление для препятствий $\underline{\widehat{S}}_{i1,i2,...,ik} \subset \underline{\widehat{S}}_{i1,i2,...,i(k-1)} \subset$ $\subset ... \subset \underline{\widehat{S}}_{i1} \subset \widehat{G}_{i1}$ и аналогичные представления для робота и эталонов.

Представление структуры препятствий двумя уровнями вложения изображено на рис. 2. Структура S содержит шесть полей.

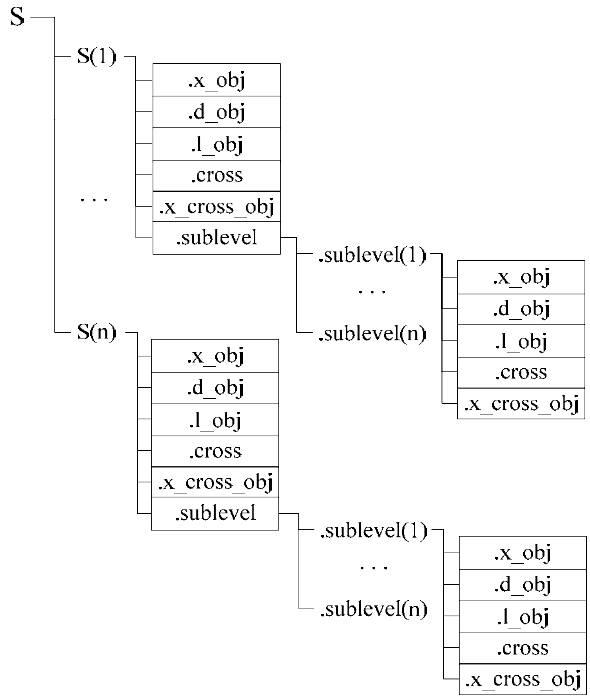


Рисунок 2 – Блок-схема структуры препятствий

1. x_obj – 3-вектор центра координат препятствия;
2. d_obj – вектор параметров препятствий (радиус шара, длина ребра параллелепипеда, радиус цилиндра);
3. l_obj – тип препятствия (1 – шар, 2 – параллелепипед, 3 – цилиндр);
4. cross – признак пересечения с элементом робота (1 – есть, 0 – нет);
5. x_cross_obj – 6-вектор координат начала и конца отрезка пересечения;
6. sublevel – вектор структур, размерность которого зависит от количества вложенных препятствий.

Оператором заполняются поля x_obj, d_obj, l_obj, sublevel, остальные поля структуры заполняются в процессе работы алгоритма.

Поиск на графе безопасных траекторий

На основании результатов построения траекторий и определения пересечений в структуру S заносятся полученные данные. Результатом отработки алгоритма формирования графа безопасных траекторий является симметричная матрица безопасных траекторий M размерности $n \times n$, где n − количество вершин графа, т.е. безопасных конфи-

гураций манипулятора. Матрица *M* строится на основе пробных траекторий, задаваемых оператором в виде начальных q^0 и конечных q^T конфигураций, поэтому вершинами графа являются свободные конфигурации (не попадающие в препятствия) пробных траекторий, задаваемых оператором. Они же отвечают диагональным элементам матрицы *M*, которые поэтому равны 1. Итак, матрица $M = \lfloor m_{ij} \rfloor$, порядка n, в которой

$$m_{ij} = \begin{cases} 1, & \text{если существует путь без пресечений из вершины } q_i \text{ в вершину } q_j, \\ -1, & \text{если путь без пересечений из вершины } q_i \text{ в вершину } q_j \text{ не существует.} \end{cases} \quad (3)$$

Безопасные траектории, соответствующие всем $m_{ij} = 1$, хранятся в массиве ячеек *MT*, то есть траектория из вершины *i* в вершину *j*, если $m_{ij} = 1$, хранится в ячейке $MT_{ij} = (q^i, ..., q^j)$, где $q^i = (q_1, q_2, ..., q_n)$ и $q^j = (q_1, q_2, ..., q_n)$, n – число обобщенных координат робота.

Если построенная из начальной точки в конечную точку каноническая траектория пересекается с препятствиями, то осуществляется поиск пути на графе безопасных траекторий методом сетевого графа. Траектория задается в виде двух точек (конфигураций): начала q^0 и конца q^T траектории. Может оказаться, что данных конфигураций на графе нет, тогда находятся ближайшие к заданным конфигурациям q^0 и q^T конфигурации на графе безопасных траекторий q^{beg} и q^{end}. Затем проверяется, есть ли пути без пересечений из q^0 в q^{beg} и из q^{end} в q^T. Если данные пути существуют, то осуществляется поиск наикратчайшего пути на графе безопасных траекторий методом Дейкстры [9] из вершины q^{beg} в вершину q^{beg}. В том случае, если путь из вершины q^{beg} в вершину q^{beg} найден, то искомая траектория без пересечений выглядит так: $Q^{trace} = (q^0, ..., q^{beg}, ..., q^{end}, ..., q^T)$.

Локальная коррекция траектории

Для локальной коррекции траектории могут использоваться разные методы, связанные с методами поиска экстремумов: случайный поиск, градиентные, нейронные сети и т.п.

Простейшим из них является случайный поиск промежуточной точки. Берём самые удалённые от начала и конца канонической траектории точки $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, ..., q_n^{beg,loc}\right)$, $q^{end,loc} = \left(q_1^{end,loc}, q_2^{end,loc}, ..., q_n^{end,loc}\right) \notin Q^u$, ещё не пересекающиеся с препятствиями, и середину отрезка, образуемого этими точками. Запускаем генератор случайных чисел для каждой присоединённой (обобщённой) координаты робота и умножаем его на d – расстояние (полуторное, удвоенное) между точками $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, ..., q_n^{beg,loc}\right)$, $q^{end,loc} = \left(q_1^{end,loc}, q_2^{end,loc}, ..., q_n^{end,loc}\right)$. Добавляем его к начальной точке $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, ..., q_n^{beg,loc}\right)$. Проверяем, что эта новая точка $q^{new,loc,1} = \left(q_1^{new,loc,1}, q_2^{new,loc,1}, ..., q_n^{new,loc,1}\right)$ попала в свободную зону, и если так, то строим каноническую траекторию в неё из начальной точки. Проверяем отсутствие пересечений этой новой подтраектории. Если траектория свободная, то включаем новую точку $q^{new,loc,1}$ в таблицу, заменяем начальную точку $q^{beg,loc}$ на новую точку $q^{new,loc,1}$ и строим траекторию из $q^{new,loc,1}$ в конечную точку $q^{end,loc}$. И так, пока не найдём новую свободную точку. Если долго не находится, то увеличиваем параметр расстояния *d* и т.д.

Для генетического алгоритма возможна следующая модификация. Вместо выбора случайного числа берём число в диапазоне между этими векторами $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, ..., q_n^{beg,loc}\right), q^{end,loc} = \left(q_1^{end,loc}, q_2^{end,loc}, ..., q_n^{end,loc}\right)$ и добавляем к начальному вектору добавки, кодируемые результатом работы функции генетического алгоритма. Например, весь диапазон отдельной конфигурационной координатой для $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, ..., q_n^{beg,loc}\right), q^{end,loc} = \left(q_1^{end,loc}, q_2^{end,loc}, ..., q_n^{end,loc}\right)$ разбиваем на 2^N частей, при этом каждый диапазон кодируется двоичным числом типа (1 0 1 0 0 0 …1 0 1) (N цифр). Кодировка для всего вектора присоединённых координат составит двоичное число размерности N x n. Это и будет числом, которое будет генерировать генетический алгоритм. Результатом работы будет некоторый код, который отвечает максимальному удалению от препятствий. Если это расстояние положительно, то эта промежуточная точка и выбирается.

Если она соединяет без пересечений с препятствиями обе точки $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, \ldots, q_n^{beg,loc}\right), q^{end,loc} = \left(q_1^{end,loc}, q_2^{end,loc}, \ldots, q_n^{end,loc}\right)$, то всё хорошо. Иначе повторяем процедуру с заменой одной из точек $q^{beg,loc} = \left(q_1^{beg,loc}, q_2^{beg,loc}, ..., q_n^{beg,loc}\right)$, $q^{end,loc} = \left(q_1^{end,loc}, q_2^{end,loc}, ..., q_n^{end,loc}\right)$ на новую промежуточную.

Градиентный метод имеет следующий вид. Пусть надо найти $q^{cross} \in Q^n$ – первое (по s) пересечение препятствий с роботом, т.е. точку касания робота и препятствия, характеризующуюся фреймом $\Phi = \left\langle N_{link}, N_{object}, \mathbf{r}_{cross}, x_{cross} \right\rangle$, где N_{link}, N_{object} – номер звена и номер препятствия этого пересечения, $\mathbf{r}_{cross}, x_{cross}$ – координаты точки пересечения в системах координат и базовой, соответственно.

$$\begin{gathered}\tilde{q} = F_{OKZ}^{-1}(\tilde{p}),\\ \tilde{p} = p(q) + \Delta p,\\ \Delta p = \mathbf{e}a_e + \tau a_\tau + \mathbf{n}a_n,\end{gathered} \tag{4}$$

где $\mathbf{e} = \mathbf{p^T} + \mathbf{p^{cross}}$ – вектор, направленный из точки касания $q^{cross} \in Q^n$, в целевую (конечную) точку q^T,

τ – вектор, направленный вдоль поверхности препятствия, т.е. по касательной,

$\mathbf{n}$ – вектор, направленный по нормали от этой поверхности,

a_e, a_τ, a_n – параметры, задающие величину шага смещения по векторам $\mathbf{e}, \tau, \mathbf{n}$.

Структура программы

Рассмотрим работу алгоритма построения свободной траектории робота на уровне связи подпрограмм (рис. 3). Прежде всего, строится каноническая траектория и определяется её пересечение с препятствиями [5]. В случае наличия пересечений ищем траекторию на графе ранее построенных безопасных траекторий. Если модуль поиска пути на этом графе не дал результатов, то осуществляется локальная коррекция траектории.

Изображены следующие подпрограммы:

- BASE – описание проблемной среды;
- LINK – математическое описание кинематики робота;
- IN_OUT_PATH – ввод/вывод данных траекторий;
- TRACE_TRAP – построение канонической траектории;

– INTERSECT_OBSTACLE_PATH – проверка пересечений робота с препятствиями;
– GRAPH_PATH – построение траекторий на графе свободных траекторий;
– LOCAL_CORRECT_PATH – локальная коррекция траектории.

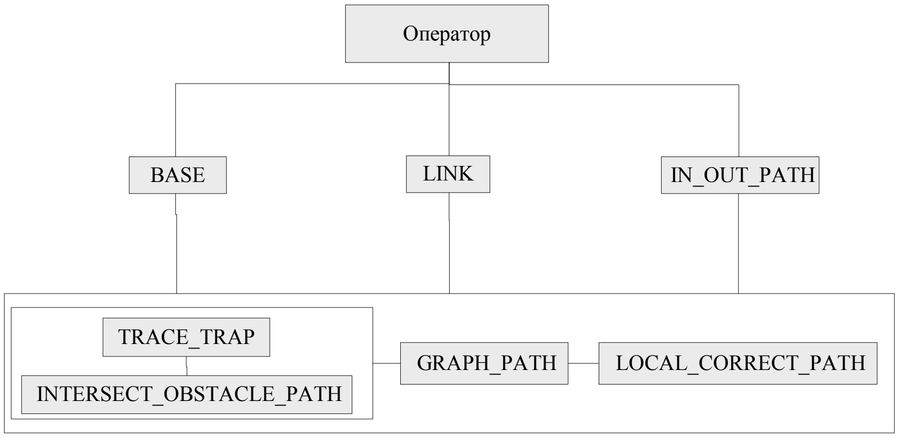


Рисунок 3 – Блок-схема программы на уровне связи основных подпрограмм

Подпрограмма TRACE_TRAP имеет следующие входные данные:
– *n* − число обобщенных координат манипулятора;
– *kcurv_trap* = 1 обычная трапеция, 2 сглаженная фильтром 1-го порядка;
– *q0* − начальная точка траектории;
– *qt* − конечная точка траектории;
– *q2_accel* − величина ускорения разгона;
– *q1_const* − величина постоянной скорости;
– *t_filter* − постоянная времени фильтра сглаживания;
– *t* − текущее время.

Её выходными данными являются:
– *qp* − величина текущих обобщенных программных координат;
– *qp1* − величина текущих обобщенных программных скоростей;
– *qp2* − величина текущих обобщенных программных ускорений.

Подпрограмма MONITIRING осуществляет поиск на графе безопасных траекторий. Её входными данными являются начальные q^0 и конечные q^T точки пробных траекторий, такие, что $q^0, q^T \in Q^n$.

Выходом подпрограммы MONITIRING является матрица безопасных траекторий *M* графа дорожной карты. Матрица *M* эквивалентна графу, на котором далее можно искать путь. Другие параметры:
– *S* – структура препятствий;
– *q0* – начальная конфигурация манипулятора;
– *qt* – конечная конфигурация манипулятора;
– *t* – текущее время;
– *q(s,j)* – обобщенная координата как функция параметра длины дуги s;
– *np* – количество конфигураций (узлов траектории);
– *cross* – признак пересечения, cross = 1 – есть, cross = 0 – нет;
– *N1* – количество препятствий на первом уровне пирамидального представления структуры пространства;
– *N2* – количество препятствий второго уровня i-го препятствия первого уровня;

– *N3* – количество препятствий третьего уровня, принадлежащее i-му препятствию первого уровня, j-му препятствию второго уровня;
– *N4* – препятствия четвертого уровня, принадлежащие i-му препятствию первого уровня, j-му препятствию второго уровня и k-му препятствию третьего уровня ;
– *N5* – препятствия пятого уровня, принадлежащие i-му препятствию первого уровня, j-му препятствию второго уровня, k-му препятствию третьего уровня, n-му препятствию четвертого уровня;
– *M* – матрица безопасных траекторий графа дорожной карты.

Алгоритм поиска на графе безопасных траекторий реализован в программе GRAPH_PATH. Входом программы GRAPH_PATH являются начальная q^0 и конечная q^T конфигурации траектории и матрица безопасных траекторий *M*. Выходом программы GRAPH_PATH является искомая траектория (q^0, q^T) без пересечений, если такая траектория существует. Если такую траекторию найти на графе не удалось, то согласно методу переменных стратегий переходим к локальной коррекции траектории в подпрограмме LOCAL_CORRECT_PATH.

Алгоритм коррекции реализован введением случайной пробной точки. Входом для программы LOCAL_CORRECT_PATH являются начальная q^0 и конечная q^T конфигурации траектории. Выходом программы является траектория (q^0, q^T) без пересечений.

Экспериментальное исследование и выводы

Оператор задаёт препятствие, заполняя древовидную структуру данных *S*. В рассматриваемом примере уровень вложения препятствий равен трём (рис. 4).

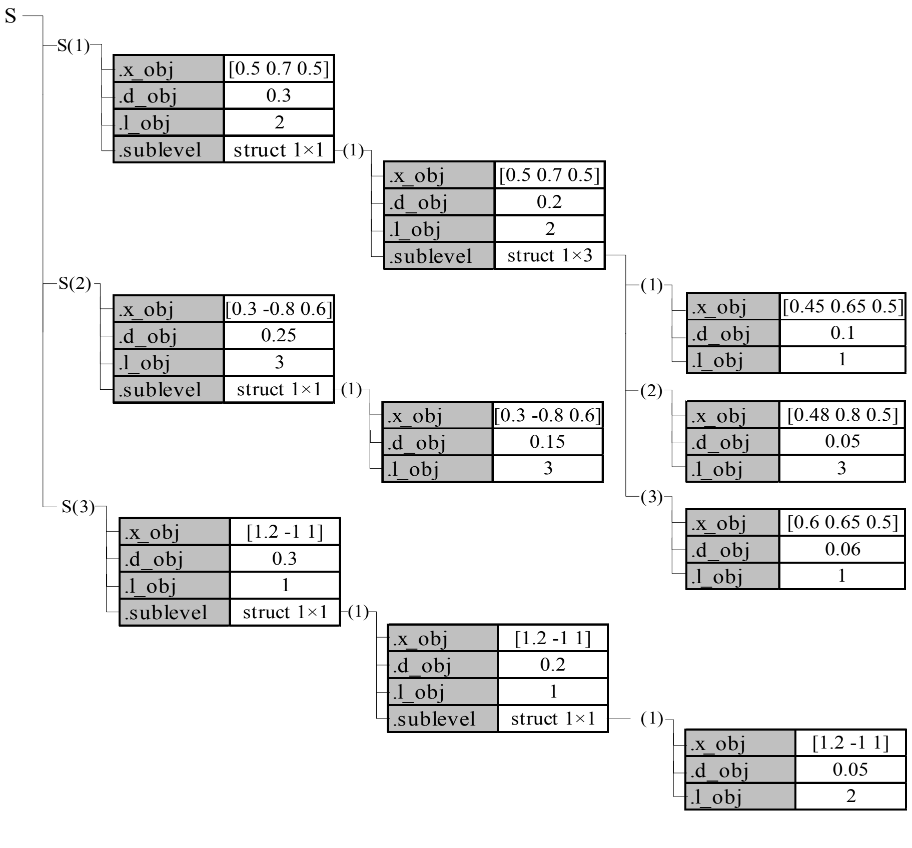


Рисунок 4 – Представление древовидной структуры *S*

Изображенное на рис. 4 представление структуры означает, что на первом уровне моделируемой проблемной среды находятся три препятствия.

1. Параллелепипед с центром *x_obj* = (0.5, 0.7, 0.5) и расстоянием от центра до граней, равным *d_obj* = 0.3, внутри которого находится препятствие второго уровня:

1.1. параллелепипед с центром *x_obj* = (0.5, 0.7, 0.5) и расстоянием от центра до граней, равным *d_obj* = 0.2, внутри которого расположены три препятствия на третьем уровне:

1.1.1. шар с центром *x_obj* = (0.45, 0.65, 0.5) и радиусом *d_obj* = 0.1м;

1.1.2. цилиндр с центром *x_obj* = (0.48, 0.8, 0.5) и радиусом *d_obj* = 0.05м;

1.1.3. шар с центром *x_obj* = (0.6, 0.65, 0.5) и радиусом *d_obj* = 0.06м.

2. Цилиндр с центром *x_obj* = (0.3, – 0.8, 0.6) и радиусом основания *d_obj* = 0.25м, внутри которого расположено препятствие на втором уровне:

2.1. цилиндр с центром *x_obj* = (0.3, – 0.8, 0.6) и радиусом основания *d_obj* = 0.15м.

3. Шар с центром *x_obj* = (1.2, – 1, 1) и радиусом *d_obj* = 0.3м, внутри которого расположено препятствие на втором уровне вложения:

3.1. шар с центром *x_obj* = (1.2, – 1, 1) и радиусом *d_obj* = 0.2м, внутри которого расположено препятствие на третьем уровне:

3.1.1. параллелепипед с центром *x_obj* = (1.2, – 1, 1) и расстоянием от центра до граней, равным *d_obj* = 0.05м.

На рис. 5 схематично показано графическое представление смоделированной проблемной среды с помощью описания проблемной среды средствами Matlab.

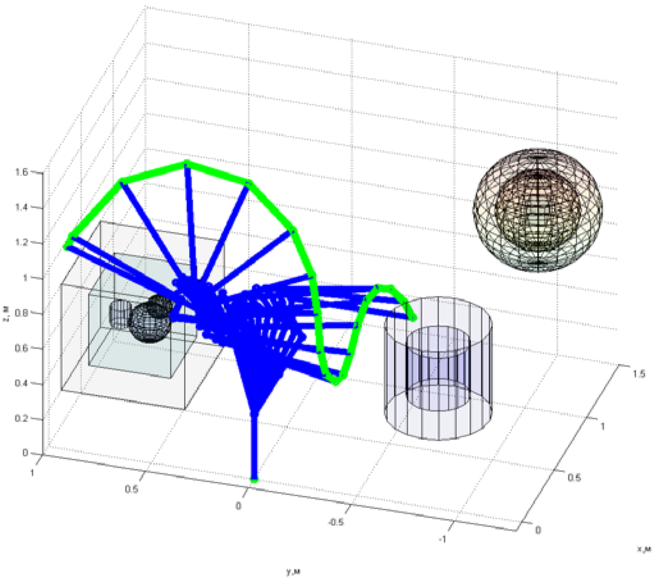


Рисунок 5 – Представление узловых конфигураций на траектории $(q^0, q^t)^0$

Таблица 1 – Узловые конфигурации траектории $(q^0, q^t)^0$

t, с	Вектор обобщенных координат, q					
0	0	0	0	0	0	0
5	0,1952	0,2362	-0,1952	0	0,1952	0
10	0,4392	0,5315	-0,4392	0	0,4392	0
15	0,6832	0,8268	-0,6832	0	0,6832	0
20	0,9272	1,1220	-0,9272	0	0,9272	0
25	1,1712	1,4173	-1,1712	0	1,1712	0
30	1,4151	1,7126	-1,4151	0	1,4151	0
35	1,5700	1,9000	-1,5700	0	1,5700	0

Рассмотрим макет шестистепенного космического манипулятора [8]. Рассмотрим построенную каноническую траекторию с началом в точке $q^0 = (0,0,0,0,0,0)$ и концом в точке $q^t = (1.57, 1.9, -1.57, 0, 1.57, 0)$. На рис. 5 и в табл. 1 приведены начальная, конечная и промежуточные конфигурации, а также траектория схвата. Траектории пересекаются с препятствием № 1 в структуре *S*. В результате отработки алгоритма проверки пересечений робота с препятствиями выявлено, что следующие промежуточные конфигурации пересекают препятствия: $q^{10} = (0.4392, 0.5315, -0.4392, 0, 0.4392, 0)$, $q^{15} = (0.6832, 0.8268, -0.6832, 0, 0.6832, 0)$, $q^{20} = (0.9272, 1.1220, -0.9272, 0, 0.9272, 0)$.

Так как траектория $(q^0, q^t)^0 = ((0,0,0,0,0,0),(1.57,1.9,-1.57,0,1.57,0))$ пересекает препятствия, то модифицируем её с помощью алгоритма локальной коррекции, что приведёт к траектории, отображённой в табл. 2 и на рис. 6 где изображена траектория схвата робота.

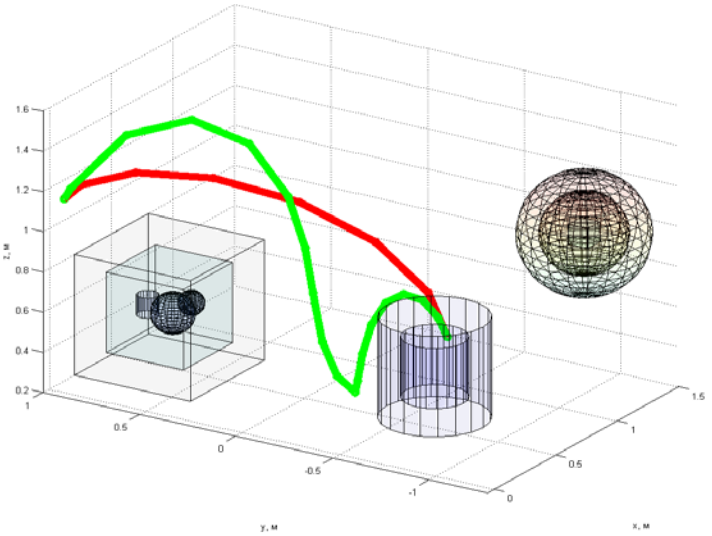


Рисунок 6 – Представление движения схвата на траекториях $(q^0, q^t)^1, (q^0, q^t)^1$

Таблица 2 – Узловые конфигурации траектории $(q^0, q^t)^1$

t, c	Вектор обобщенных координат, q					
0	0	0	0	0	0	0
5	0,0456	0,0570	-0,0471	0,3986	0,0471	0
10	0,1026	0,1282	-0,1059	0,8968	0,1059	0
15	0,1595	0,1994	-0,1648	1,3950	0,1648	0
20	0,2165	0,2706	-0,2236	1,8933	0,2236	0
25	0,2735	0,3419	-0,2825	2,3915	0,2825	0
30	0,3305	0,4131	-0,3413	2,8897	0,3413	0
35	0,3874	0,4843	-0,4002	3,3880	0,4002	0
40	0,4283	0,5354	-0,4424	3,7453	0,4424	0
45	0,5351	0,6689	-0,4943	3,3789	0,5527	0
50	0,6686	0,8358	-0,5593	2,9209	0,6906	0
55	0,8021	1,0026	-0,6242	2,4630	0,8285	0
60	0,9074	1,1342	-0,6754	2,1018	0,9372	0
65	1,0068	1,2585	-0,8206	1,7608	1,0399	0
70	1,1311	1,4138	-1,0020	1,3345	1,1682	0
75	1,2553	1,5691	-1,1835	0,9082	1,2966	0
80	1,3796	1,7244	-1,3649	0,4819	1,4249	0
85	1,5023	1,8779	-1,5442	0,0606	1,5518	0

По результатам эксперимента можно сделать вывод, что траектория, построенная с помощью алгоритма локальной коррекции, может получиться не вполне гладкой и чаще всего не оптимальной. Для получения лучших результатов нужно использовать другие алгоритмы локальной коррекции. Однако следует отметить, что реализованный метод локальной коррекции логически прозрачен, математически прост и не требует больших вычислительных затрат.

Кроме рассмотренного простейшего примера был построен ещё ряд траекторий, в том числе, с использованием графа безопасных траекторий. Таким образом, метод показал свою работоспособность и эффективность.

Литература

1. Latombe J. Robot Motion Planning. – Kluwer, Boston, MA,1991.
2. LaValle S.M. Planning Algorithms [Электронный ресурс]: 2004. – 859 с. – Режим доступа: http://msl.cs.uiuc.edu/planning/
3. Вирт Н. Алгоритмы и структуры данных. – М.: Невский Диалект, 2006. – С. 352.
4. Kavraki L. and Latombe J.-C. Randomized preprocessing of configuration space for fast path planning // In Proc. IEEE Int. Conf. Robotics and Automation. – San Diego, CA. – 1994. – P. 2138-2145.
5. Kavraki L. E., Latombe J.C. Probabilistic Roadmaps for Robot Path Planning, Robotics Laboratory, Department of Computer Science Stanford University. – Stanford, California CA 94305. – 1998. – 21 p.
6. Макарычев В.П. Методы управления космическими роботами // «Искусственный Интеллект». – 2003. – № 4. – С. 140-147.
7. Макарычев В.П. Использование интеллектуальных технологий при построении траекторий роботов в среде с препятствиями // Материалы Международной научной конференции Искусственный интеллект. Интеллектуальные и многопроцессорные системы-2004. – Т. 2. Таганрог: Изд-во ТРТУ. – 2004. – С. 401-405.
8. Макарычев В.П., Юревич Е.И. Супервизорное управление космическими манипуляторами. – СПб.: Астерион, 2005. – 108 с.
9. Dijkstra E.W. A note on two problems in connection with graphs // Numerische Mathematik. – 1959. – V. 1. – P. 269-271.
10. Динамика управления роботами / В.В. Козлов, В.П. Макарычев, А.В. Тимофеев, Е.И. Юревич / Под ред. Е.И. Юревича. – М.: Наука. Главная редакция физико-математической литературы, 1984. – 336 с.

V.P. Makarychev

The Variable Strategy Method of Constructing Trajectories of the Motion of Robot in the Environment with Obstacles

In this paper is proposed and investigated the method of constructing trajectories of the motion of robots in environments with obstacles based on the change of behavior strategies depending of situation. Method consists of a number of steps. First is made an attempt of the construction into the purposeful point of canonical trajectory in the free space, optimum for the time and taking into account dynamic limitations. The rapid analysis of the presence of intersections of the constructed trajectory with obstacles is based on their pyramidal structure. Are described the realizing the developed method computer program at Matlab and results of experimental studies based on the example of the space technological manipulator.

Статья поступила в редакцию 22.07.2008.